YOU ONLY EDIT ONCE: INCENTIVIZING IN-CONTEXT CAPABILITY OF LLMS VIA LOCAL DEMONSTRATION REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning (ICL) is crucial for boosting the inference performance of large language models (LLMs). However, the effectiveness of ICL in LLMs is greatly influenced by the choice of demonstration sets. Exhaustive searches over these sets are combinatorial, and existing selectors often rely on relevance or likelihood proxies to implicitly assess ICL quality. Making repeated queries to the target LLM with these strategies can incur substantial costs. This work simplifies selection by framing it as a constrained local search problem and presents local demonstration editing (LDE). Starting with an initially retrieved set of demonstrations, LDE employs a single structured edit to explore its surrounding neighborhood while balancing performance gains with search costs. Technically, LDE is reduced to a policy search problem, for which we train a small LLM, referred to as Jev-LDE. This model as the System-1 modifies the retrieved demonstration set by performing actions such as `Keep`, `Delete`, or `Replace` elements, all within a framework of reinforcement learning with verifiable rewards. At test time, Jev-LDE executes a single edit of the retrieved demonstration set, followed by one inference from the target LLM, avoiding the need for iterative context scoring or subset searches. Across standard classification benchmarks, various target LLMs with Jev-LDE as the plug-and-play module consistently improve ICL performance, and Jev-LDE shows transferability to held-out benchmarks and models without retraining. These findings indicate that the LDE approach offers an efficient and adaptable method for harnessing the ICL capabilities of target LLMs.

1 INTRODUCTION

Large language models (LLMs) possess the ability to adapt to new tasks during inference by conditioning on a set of demonstrations, without necessitating any updates to their parameters (Min et al., 2022). This phenomenon is referred to as in-context learning (ICL) (Brown et al., 2020), which is particularly beneficial when LLMs are used as frozen or black-box systems to serve a diverse collection of users and tackle distinct tasks.

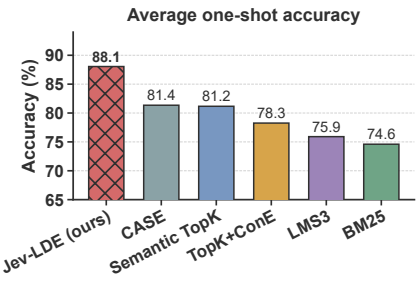


Figure 1: **One-shot classification result.** The accuracy is averaged over TREC, DBPedia, and Banking77 using four target LLMs.

More Examples, Not Always Better in ICL. However, this paradigm's effectiveness depends heavily on the quality and quantity of demonstration examples attached to the prompt. Note that inference is often constrained by memory and computational resources (Jiang et al., 2023; Kwon et al., 2023), restricting the number of demonstrations that fit within the context window. The relationship between context length and prediction accuracy is not always linear; longer contexts do not necessarily bring better predictions (Liu et al., 2024). Irrelevant or repetitive examples can hurt performance by consuming valuable tokens and diverting attentional resources (Shi et al., 2023; Gupta et al., 2023). Hence, specifying an opti-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

mal subset of examples that improves the target LLM's accuracy under realistic constraints poses a pressing demand in few-shot ICL.

One Edit Can Make a Big Difference. Existing demonstration selection methods screen candidates from heuristic signals implicitly related to answer correctness. Semantic retrieval ranks by embedding similarity (Liu et al., 2022), while BM25 ranks by lexical overlap (Robertson & Zaragoza, 2009). Model-aware methods instead score candidates with the target's own signals, e.g., generator likelihood (Rubin et al., 2022), cross-entropy difference (Iter et al., 2023), conditional entropy (Peng et al., 2024), or exploration rewards (Purohit et al., 2025; Wang et al., 2025). The vanilla subset selection for k-shot ICL is combinatorial: evaluating all $\binom{n}{k}$ subsets of an n-candidate pool with model-aware scoring requires repeated target LLM queries, and the selection often sacrifices search efficiency for performance gains. As illustrated in Fig. 2, this trade-off need not be resolved by global search. A single membership edit to an initially retrieved context can boost overall ICL accuracy, implying that the significant potential improvement resides within the one-edit neighborhood.

Naturally, the above drives us to answer two research questions (RQs):

1. Can we develop a scalable schema to edit the retrieved demonstration just once in order to effectively enhance ICL performance?
2. Does the trained editor illustrate effective generalization across different configurations, including unseen benchmarks and target models?

Local Editing as a Practical Alternative. In response to these RQs, this work presents *local demonstration editing* (LDE) as a practical scheme that refines a retrieved context with one structured membership edit instead of searching over subsets. However, seeking a beneficial local edit is nontrivial. (i) The significance of an edit relies on the specific query. That is, the informative shots to improve prediction vary with each query (Rubin et al., 2022). (ii) The one-edit neighborhood is small relative to global subsets but still too large to enumerate with the target LLM's calls at test time. Some methods, like TopK+ConE (Peng et al., 2024), require repeated queries to the target LLM. Distinguished from all previous studies, this work concentrates on a special setup that allows editing the membership of an existing selector's output only once. This makes LDE compatible with typical demonstration pre-selection methods, e.g., BM25 (Robertson & Zaragoza, 2009).

Technically, we fine-tune a small LLM, called Jev-LDE, with reinforcement learning (RL) to modify the retrieved context with one atomic action: `Keep`, `Delete`, or `Replace`. This process relies on a binary reward signal obtained from a frozen target LLM's in-context prediction result, guiding the local edit decisions toward improved target-model predictions. The resulting Jev-LDE is semantic-aware and works as System-1 (Kahneman, 2011) for more effective ICL. At test time, our method starts with the initially retrieved set, edits it with Jev-LDE, and then performs in-context inference with the refined context on target LLMs.

Contributions and Empirical Findings: This work focuses on few-shot ICL scenarios and the primary contributions are twofold:

1. This work presents LDE as a constrained demonstration subset search problem that aims to incentivize the ICL capability of LLMs through a single structured membership edit, trading off performance gains and search efficiency.
2. We reduce the LDE to a control problem and fine-tune Jev-LDE via reinforcement learning as a lightweight, plug-and-play System-1 module. This module enables a fast single-context edit, better serving the target LLM's few-shot ICL at test time.

Experiments on typical classification benchmarks and mainstream target LLMs, across various shot budgets, positively answer the mentioned RQs: Jev-LDE achieves the best or tied-best performance across most configurations, e.g., a rise in one-shot accuracy from 81.2% with Semantic TopK to 88.1% (see Fig. 1). Moreover, Jev-LDE exhibits effective edit capability on setups with other held-out benchmarks and diverse target LLMs, improving few-shot test-time scaling without retraining.

2 From Combinatorial Shot Selection to Naive Local Search

2.1 Demonstration Selection Setup

Definition 1 (k-Shot Demonstration Selection)**.** Let $\mathcal{C} = \{\boldsymbol{z}_i = (\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ be a set of available demonstrations. Given a query $\boldsymbol{x}$, demonstrations $\mathcal{Z}$, the shot budget k, and $\mathcal{S}_k(\mathcal{C}) = \{\mathcal{Z} \subseteq \mathcal{C} : |\mathcal{Z}| = k\}$, the ideal selection solves

$$\mathcal{Z}_k^\star(\boldsymbol{x}) \in \arg\max_{\mathcal{Z} \in \mathcal{S}_k(\mathcal{C})} \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}, \mathcal{Z}), \tag{1}$$

which aims to maximize the likelihood of the gold answer $\boldsymbol{y}$ given the target LLM $p_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x}, \mathcal{Z})$. Since this objective is intractable at test time, a selector g provides a tractable approximation: $\mathcal{Z}_0 = g(\mathcal{C}, k, \boldsymbol{x}) \in \mathcal{S}_k(\mathcal{C})$.

Eq. (1) is a combinatorial optimization problem with an unobservable objective. The set $\mathcal{S}_k(\mathcal{C})$ consists of $\binom{n}{k}$ subsets that can only be scored through target queries. Also, the contribution of each demonstration shot depends on its context. Since target values $\boldsymbol{y}$ are unknown at test time, traditional evaluations of $\ln p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}, \mathcal{Z})$ are infeasible, driving selectors to use proxy scores like embedding similarity in Semantic TopK (Liu et al., 2022).

2.2 One Local Edit Moves Accuracy

Definition 2 (One-Edit Neighborhood)**.** For the initially retrieved set $\mathcal{Z}_0$ and demonstration set $\mathcal{Z}$, let the Replacement distance to the retrieved initialization be $d(\mathcal{Z}, \mathcal{Z}_0) = k - |\mathcal{Z} \cap \mathcal{Z}_0|$, then the one-edit neighborhood of $\mathcal{Z}_0$ is $\{\mathcal{Z} | d(\mathcal{Z}, \mathcal{Z}_0) \leq 1, \mathcal{Z} \in \mathcal{S}_k(\mathcal{C}) \cup \mathcal{S}_{k-1}(\mathcal{C})\}$.

This part quantitatively analyzes how much accuracy hinges on a single membership change. It can be viewed as a constrained version of Eq. (1). Obviously, the neighborhood reflects three probable cases: $d = 0$ Keeps $\mathcal{Z}_0$, while $d = 1$ removes one retrieved demonstration ($|\mathcal{Z}| = k - 1$) or swaps it for a candidate in $\mathcal{C} \setminus \mathcal{Z}_0$ ($|\mathcal{Z}| = k$).

One Smart Edit Unlocks Massive Gains. On TREC test queries (Li & Roth, 2002), we evaluate all valid actions for a 4-shot setup. As Fig. 2 shows, choosing the best action per query using gold labels yields gains of 16.8 and 26.0 percentage points over the reported Semantic TopK baselines for Qwen2.5-7B and Llama-3-8B (Qwen Team, 2024; Llama Team, 2024), respectively. Even though these oracle approaches demonstrate a high potential for performance improvement, Jev-LDE still brings significant gains of 7.4 and 8.2 points even without the use of test labels. In contrast, uniformly random actions hardly yield the average performance gain. Thus, substantial improvement is available within $d(\mathcal{Z}, \mathcal{Z}_0) \leq 1$, motivating a policy that learns to identify beneficial one-step edits from the target LLM's feedback.

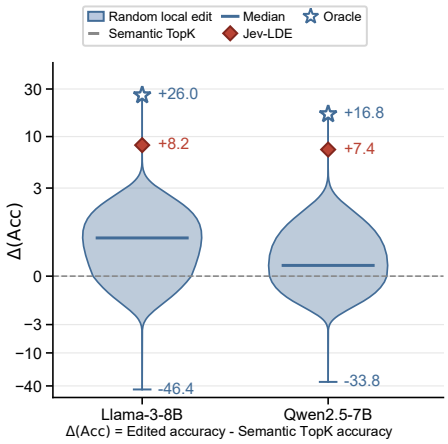


Figure 2: **One edit can substantially affect 4-shot TREC accuracy.** Violins show the full accuracy-change distribution under uniformly sampled valid actions. The Oracle using test labels to select the optimal subset.

3 Learn to Edit Locally with Reinforcement Learning

3.1 Local Editing Formulation

The problem statement of LDE is rooted in Definition 1, but narrows its feasible set to the retrieved initialization $\mathcal{Z}_0$ along with its one-edit neighborhood $d(\mathcal{Z}, \mathcal{Z}_0)$. This corresponds to:

$$\hat{\mathcal{Z}}^\star(\boldsymbol{x}) \in \arg\max_{\hat{\mathcal{Z}} \in \mathcal{S}_k(\mathcal{C}) \cup \mathcal{S}_{k-1}(\mathcal{C}):\ d(\hat{\mathcal{Z}}, \mathcal{Z}_0) \leq 1} \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}, \hat{\mathcal{Z}}), \tag{2}$$

where LDE performs a local search over the combinatorial problem specific to each query, seeking a set that improves on $\mathcal{Z}_0$.

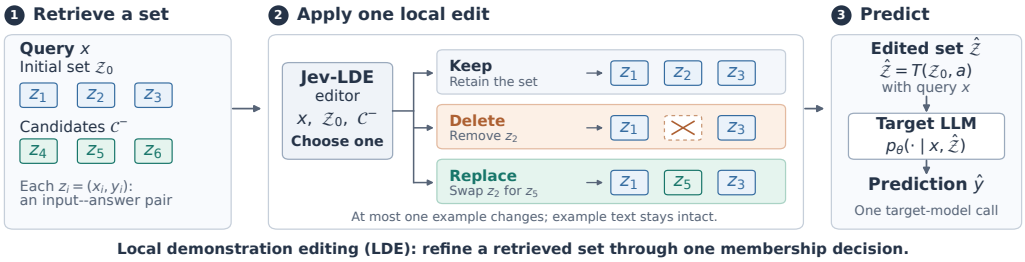


Figure 3: **Overview of local demonstration editing (LDE).** Given a initially retrieved set $\mathcal{Z}_0$, e.g., using Semantic TopK, and candidates $\mathcal{C}^-$, Jev-LDE chooses one action to edit $\mathcal{Z}_0$, then edited set supports one frozen-target prediction.

Reducing Local Editing to Policy Optimization. This work approximates Eq. (2) through fine-tuning a small LLM, Jev-LDE (See Fig. 3). As a lightweight System-1 module, Jev-LDE learns a stochastic editing policy that captures the semantic relationships between x and $\mathcal{Z}_0$, serving *fast decision-making purposes* and enhancing the target LLM's in-context predictions.

Note that several edits can yield equally positive outcomes, hence RL offers a natural framework for balancing exploration of diverse edits with exploitation of rewarding choices. Training uses labeled ICL training or validation examples, with the frozen target LLM $p_{\boldsymbol{\theta}}$ providing verifiable rewards, i.e., post-edited prediction accuracy. At test time, Jev-LDE directly edits the initially retrieved set to create input for target LLM inference.

Combinatorial Search under a Single-Edit Constraint. For a query $\boldsymbol{x}$ with candidate pool $\mathcal{C}$ and initially retrieved set $\mathcal{Z}_0$ from a specific pre-selector (e.g., Semantic TopK), let $[k] = \{1, \ldots, k\}$ index the demonstrations in $\mathcal{Z}_0$, and let $\mathcal{C}^- = \mathcal{C} \setminus \mathcal{Z}_0$ denote the remainder set. Then the action space for Jev-LDE is

$$\mathcal{A}_{\text{LDE}}(\mathcal{Z}_0, \mathcal{C}^-) = \{\texttt{Keep}\} \cup \{\texttt{Delete}(j) : j \in [k]\} \cup \{\texttt{Replace}(j, z) : j \in [k],\ z \in \mathcal{C}^-\}, \quad (3)$$

where `Keep` retains $\mathcal{Z}_0$, $\texttt{Delete}(j)$ removes its j-th demonstration, and $\texttt{Replace}(j, z)$ swaps that demonstration for a candidate $z \in \mathcal{C}^-$.

All feasible actions in Eq. (3) induce the corresponding post-edited demonstration set in Eq. (2), constituting the one-edit neighborhood of $\mathcal{Z}_0$.

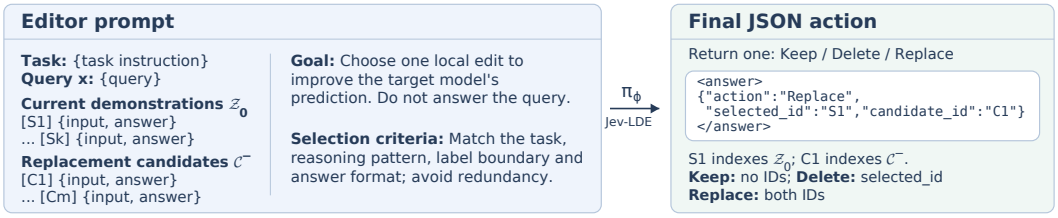


Figure 4: **Jev-LDE editor interface.** Condensed prompt and JSON action format, illustrated with a `Replace` action. The full template appears in Appendix C.

3.2 Training with Target-Model Feedback

As displayed in Fig. 4, Jev-LDE is in the form of a small LLM editor $\pi_{\boldsymbol{\phi}}(\cdot|s)$. Its input s contains the task instruction, query $\boldsymbol{x}$, initially retrieved set $\mathcal{Z}_0$, and Replacement candidates $\mathcal{C}^-$. It returns one `Keep`,`Delete`, or `Replace` action as JSON inside `<answer>` tags, referencing demonstrations by identifier. Invalid outputs receive zero reward during post-training and fall back to `Keep` during evaluation (See template details in Appendix C).

Verifiable Reward Design. For the example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, let s_i be the editor prompt constructed using $\mathcal{Z}_{0,i}$ and $\mathcal{C}_i^-$. A sampled completion $o_{i,g} \sim \pi_{\boldsymbol{\phi}_{\text{old}}}(\cdot|s_i)$ specifies an action $a_{i,g}$, which yields

$\mathcal{Z}_{i,g} = T(\mathcal{Z}_{0,i}, a_{i,g})$. The frozen target LLM $p_{\boldsymbol{\theta}}$ then produces one prediction $\hat{\boldsymbol{y}}_{i,g}$ from $(\boldsymbol{x}_i, \mathcal{Z}_{i,g})$. The reward function $r_{i,g} = \mathbb{I}[\hat{\boldsymbol{y}}_{i,g} = \boldsymbol{y}_i]$ follows the task-specific answer-matching rule, directly receiving edit feedback through target-LLM inference results (Wang et al., 2025; Shao et al., 2024; DeepSeek-AI, 2025).

Optimization Objective. For each prompt s_i, we sample G completions from Jev-LDE and compute the reward mean μ_i and standard deviation σ_i. The normalized advantage is $\hat{A}_{i,g} = \frac{r_{i,g}-\mu_i}{\sigma_i+\epsilon_{\text{adv}}}$, with $\epsilon_{\text{adv}} > 0$ to ensure numerical stability. For a minibatch $\mathcal{B}$, we maximize the clipped group relative policy optimization (GRPO) objective (Shao et al., 2024; Yu et al., 2025):

$$\mathcal{J}_{\text{GRPO}}(\boldsymbol{\phi}) = \frac{1}{|\mathcal{B}|G} \sum_{i\in\mathcal{B}} \sum_{g=1}^{G} \frac{1}{T_{i,g}} \sum_{t=1}^{T_{i,g}} \min\left[\rho_{i,g,t}(\boldsymbol{\phi})\hat{A}_{i,g}, \text{clip}(\rho_{i,g,t}(\boldsymbol{\phi}), 1-\epsilon_{\text{lo}}, 1+\epsilon_{\text{hi}})\,\hat{A}_{i,g}\right].$$

Here, $\rho_{i,g,t}(\boldsymbol{\phi})$ is the current-to-old policy likelihood ratio for token t, conditioned on s_i and preceding completion tokens. The completion length is $T_{i,g}$, with ϵ_{lo} and ϵ_{hi} to control clipping. Note that the target LLM $p_{\boldsymbol{\theta}}$ is kept frozen when optimizing $\boldsymbol{\phi}$.

Inference Procedure at Test-Time. For a new query $\boldsymbol{x}$, the pre-selector formulates $\mathcal{Z}_0$ and $\mathcal{C}^-$. With the input s, Jev-LDE samples one completion $o \sim \pi_{\boldsymbol{\phi}}(\cdot|s)$, which is parsed into an action $\hat{a} \in \mathcal{A}_{\text{LDE}}(\mathcal{Z}_0, \mathcal{C}^-)$. With the edited demonstration set $\hat{\mathcal{Z}} = T(\mathcal{Z}_0, \hat{a})$ as the input, and the target LLM derives the prediction $\hat{\boldsymbol{y}} \sim p_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x}, \hat{\mathcal{Z}})$.

3.3 Implementation and Analysis

Implementation Details. In our implementation, the pre-selector g ranks the benchmark-level demonstration pool, then top-N demonstrations form the candidate pool $\mathcal{C}$, and the top-k initialize $\mathcal{Z}_0$. Algorithm 1 outlines procedures. Jev-LDE undergoes post-training with a static set of queries from several source benchmarks, using a fixed target LLM. The post-trained Jev-LDE then seamlessly serves target LLMs across benchmarks without retraining.

Algorithm 1: Jev-LDE: One-Edit Inference and Post-Training with RL

Input: query $\boldsymbol{x}$; demonstration pool $\mathcal{C}$; pre-selector g; pool size $N \geq k$; shot budget k; editor $\pi_{\boldsymbol{\phi}}$; frozen target $p_{\boldsymbol{\theta}}$; prepared training set $\mathcal{D}$; group size G; epochs E.

Output: inference: $(\hat{\mathcal{Z}}, \hat{\boldsymbol{y}})$; post-training: updated $\pi_{\boldsymbol{\phi}}$.

```
// Inference
Z_0 ← g(C, k, x)
// editing pool (top-N)
C ← g(C, N, x)
C^- ← C \ Z_0
Build s from task instruction, x, Z_0, C^-
// one completion, temperature 0
Generate o with π_φ(·|s)
â ← Parse(o)
// validate JSON and example IDs
if â is invalid then
    â ← Keep
// apply one edit
Ẑ ← T(Z_0, â)
// one prediction, temperature 0
Predict ŷ with p_θ(·|x, Ẑ)
return ŷ
```

```
// Post-training
Initialize π_φ; freeze p_θ
for e = 1 to E do
    foreach (x_i, y_i) ∈ D do
        Build Z_{0,i}, C_i^-, s_i; φ_old ← φ
        Sample {o_{i,g}}_{g=1}^G ~ π_{φ_old}(·|s_i)
        for g = 1 to G do
            a_{i,g} ← Parse(o_{i,g})
            if a_{i,g} is invalid then
                r_{i,g} ← 0
                continue
            Z_{i,g} ← T(Z_{0,i}, a_{i,g})
            Predict ŷ_{i,g} with p_θ(·|x_i, Z_{i,g})
            r_{i,g} ← I[ŷ_{i,g} = y_i]
        Compute group reward mean μ_i and std. σ_i
        Â_{i,g} ← (r_{i,g} − μ_i)/(σ_i + ε_adv) for all g
        Update φ to maximize J_GRPO(φ)
```

Theoretical Analysis. Let $r(\boldsymbol{x}, \mathcal{Z}) \in \{0, 1\}$ denote the frozen target's feedback reward. We analyze two complementary aspects of learning local edits from correctness feedback: the availability of nearby successful edits, and the feedback sufficient for learning an editing policy.

Definition 3 (Optimal Replacement distance)**.** For $\mathcal{Z} \in \mathcal{S}_k(\mathcal{C})$, the minimum Replacement distance to a correct context is $D = \min\{d(\mathcal{Z}, \mathcal{Z}_0) : \mathcal{Z} \in \mathcal{S}_k(\mathcal{C}),\ r(\boldsymbol{x}, \mathcal{Z}) = 1\}$, with $\min \varnothing = \infty$, where d is the Replacement distance defined in Section 2.2.

Proposition 1 (Geometric tail of optimal edit distance)**.** *Consider odd-k majority voting over context-invariant binary query-label judgments. Over random evaluation states, assume each retrieved judgment is correct with conditional probability at least $\alpha \in (0.5, 1)$ given the correctness of every other candidate in $\mathcal{C}$, for every conditioning pattern with positive probability. If $\Pr(0 < D < \infty) > 0$, then*

$$\Pr(D > h | 0 < D < \infty) \leq \left(\frac{1-\alpha}{\alpha}\right)^h, \qquad h = 1, 2, \ldots.$$

The event $0 < D < \infty$ denotes retrieval failures repairable by global selection from the same pool; taking $h = 1$ gives single-Replacement coverage of at least $\frac{2\alpha-1}{\alpha}$, allowing dependent judgments under the stated condition. This majority-vote model is the classical Condorcet jury theorem setting (Grofman et al., 1983) and reappears at inference time as self-consistency (Wang et al., 2023). We next turn to the feedback sufficient for learning such a policy.

Definition 4 (Policy value)**.** For a normalized editing policy $\pi_{\boldsymbol{\phi}}(a|s)$ over valid actions, define $\mathbb{E}_{\pi_{\boldsymbol{\phi}}}[r] = \mathbb{E}[r(\boldsymbol{x}, T(\mathcal{Z}_0, a))]$, where the expectation is over a fresh labeled query from the evaluation distribution, its editing state s, and $a \sim \pi_{\boldsymbol{\phi}}(\cdot|s)$.

Proposition 2 (Feedback complexity of editing)**.** *Let $\mathcal{P}$ be a nonempty finite parameter set fixed independently of training data and indexing policies as in Definition 4. For each of $|\mathcal{D}|$ independent labeled queries drawn from the evaluation distribution, independently sample a valid action $a_i \in \mathcal{A}_{\mathrm{LDE}}$ uniformly and observe its correctness reward r_i. Choose $\widehat{\boldsymbol{\phi}}$ to maximize $\sum_i r_i \pi_{\boldsymbol{\phi}}(a_i|s_i)$ over $\mathcal{P}$. For $\varepsilon, \delta \in (0,1)$, if $|\mathcal{D}| \geq 16|\mathcal{A}_{\mathrm{LDE}}|\varepsilon^{-2}\log(\frac{2|\mathcal{P}|}{\delta})$, then, with probability at least $1-\delta$,*

$$\max_{\boldsymbol{\phi} \in \mathcal{P}} \mathbb{E}_{\pi_{\boldsymbol{\phi}}}[r] - \mathbb{E}_{\pi_{\widehat{\boldsymbol{\phi}}}}[r] \leq \varepsilon.$$

Thus, the selected policy is within ε of the best policy indexed by $\mathcal{P}$ in expected correctness reward.

With policy-class size, accuracy tolerance, and confidence fixed, the sufficient feedback bound scales linearly with the number of available actions, providing a learning-theoretic rationale for local editing. This guarantee concerns the finite-class procedure in Proposition 2.

4 Experiments

This section empirically answers the RQs outlined in Section 1. We assess the advantages of Jev-LDE's editing capability across a range of scenarios and evaluate its transferability to both unseen benchmarks and target LLMs. See detailed experimental configurations in Appendix A.

Benchmarks and Backbones. Jev-LDE uses Qwen3-1.7B (Qwen Team, 2025) and is RL-post-trained on AGNews (Zhang et al., 2015), TREC (Li & Roth, 2002), DBPedia (Zhang et al., 2015), and GSM8K (Cobbe et al., 2021). Semantic TopK is the default pre-selector during training and evaluation; frozen Meta-Llama-3-8B-Instruct (Llama Team, 2024) is the sole feedback provider. Held-out benchmarks comprise Banking77 (Casanueva et al., 2020), SST-2 (Socher et al., 2013), AQuA-RAT (Ling et al., 2017), and TACRED (Zhang et al., 2017); TREC-50 is near-domain, sharing questions and the training split with TREC. Main comparisons span question, topic, and intent classification on TREC, DBPedia, and Banking77, respectively, with Qwen2.5-7B-Instruct (Qwen Team, 2024), Meta-Llama-3-8B-Instruct, Qwen3-8B (Qwen Team, 2025), and Qwen3.5-9B (Qwen Team, 2026) as target LLMs. We further test these with DeepSeek-V4-Flash (DeepSeek-AI, 2026) on TREC-50 and TACRED.

Baselines. We compare Jev-LDE against five demonstration selection methods: (i) **Semantic TopK** selects demonstrations by query–example embedding similarity (Liu et al., 2022). (ii) **BM25** ranks demonstrations by lexical relevance (Robertson & Zaragoza, 2009). (iii) **TopK+ConE** initially retrieves a Top-30 semantic pool and Keeps the k candidates with the best conditional-entropy scores (Peng et al., 2024). (iv) **LMS3** combines similarity and inference stability (Liu et al., 2025). (v) **CASE** performs sample-efficient subset selection (Purohit et al., 2025).

Evaluation protocol. Similar to LMS3 (Liu et al., 2025), all methods are evaluated with shot budgets $k \in \{1, 2, 3, 4\}$. For Jev-LDE, the initial k-shot set is constructed by Semantic TopK. All methods use the same data splits, prompts, and answer-normalization procedure for fair comparison. The decoding temperature is set to zero for both Jev-LDE and target LLMs during evaluation to circumvent sampling randomness.

Table 1: **Few-shot classification accuracy (%).** All results are averaged over five runs. Best results, including ties, are bold; second-best results are underlined. Blue rows denote Jev-LDE.

Target model	Method	TREC				DBPedia				Banking77			
		1-shot	2-shot	3-shot	4-shot	1-shot	2-shot	3-shot	4-shot	1-shot	2-shot	3-shot	4-shot
Qwen2.5-7B-Instruct	Random	16.8	27.1	35.2	40.5	33.0	39.4	43.5	48.5	4.3	5.5	6.4	7.6
	Semantic TopK	65.4	75.0	79.0	80.4	89.7	93.9	95.0	95.3	88.6	90.7	90.8	91.6
	BM25	64.6	77.6	81.6	84.4	86.4	91.7	94.1	95.4	75.0	80.8	83.6	85.0
	TopK+ConE	46.6	57.2	60.6	65.6	82.2	90.1	92.3	93.7	70.8	78.5	81.0	82.8
	LMS3	65.6	75.0	79.2	80.8	89.5	93.7	95.0	95.1	**88.7**	90.8	90.8	91.3
	CASE	65.6	75.4	79.2	81.8	89.7	93.8	95.2	95.3	**88.7**	90.7	90.9	91.5
	Jev-LDE (ours)	**80.6**	**86.0**	**88.4**	**87.8**	**94.8**	**95.6**	**96.1**	**96.5**	88.4	**91.5**	**91.8**	**92.3**
Meta-Llama-3-8B-Instruct	Random	8.2	17.7	24.4	31.1	27.7	28.6	30.5	30.2	3.1	5.1	6.1	7.0
	Semantic TopK	52.8	71.6	71.8	71.4	86.1	91.8	91.7	89.9	91.1	92.0	92.2	91.9
	BM25	45.6	68.4	69.0	70.6	84.9	89.5	89.7	88.3	78.0	82.1	84.2	85.4
	TopK+ConE	55.2	72.2	72.0	74.0	89.1	94.5	94.4	95.0	89.7	90.8	91.2	91.5
	LMS3	35.8	62.8	65.8	66.6	76.8	85.7	88.7	87.3	79.8	84.3	85.3	85.8
	CASE	52.6	72.0	72.6	71.2	86.1	92.0	91.7	90.1	**92.2**	**92.1**	92.2	92.0
	Jev-LDE (ours)	**65.6**	**81.4**	**81.8**	**79.6**	**93.0**	**96.0**	**95.7**	**95.6**	91.6	91.9	**92.6**	**92.2**
Qwen3-8B	Random	24.1	30.4	36.2	41.7	29.1	33.7	38.0	41.9	4.4	6.1	7.5	8.9
	Semantic TopK	67.0	77.0	81.4	84.0	90.0	95.2	96.4	96.8	89.8	92.0	91.9	92.1
	BM25	63.2	77.8	83.0	86.6	86.1	92.4	94.3	96.0	76.8	83.3	85.5	86.9
	TopK+ConE	65.2	76.2	82.0	83.4	89.7	95.6	96.4	97.4	88.4	91.5	91.8	91.7
	LMS3	61.2	73.4	81.2	83.6	84.8	93.0	94.9	96.2	75.3	83.7	86.8	88.4
	CASE	67.2	78.0	80.4	85.2	90.2	94.9	96.4	96.9	89.9	91.9	92.0	92.1
	Jev-LDE (ours)	**81.4**	**86.8**	**89.4**	**89.2**	**95.3**	**96.7**	**97.1**	**97.6**	**90.3**	**92.7**	**92.6**	**93.5**
Qwen3.5-9B	Random	27.4	40.3	49.6	56.5	25.7	31.6	37.1	41.0	5.3	7.4	8.9	10.1
	Semantic TopK	72.0	84.0	87.4	89.4	90.3	94.8	96.5	96.6	91.2	92.3	92.6	92.8
	BM25	70.4	82.2	86.0	89.2	85.9	92.1	94.9	96.8	78.3	83.3	86.4	87.4
	TopK+ConE	79.2	88.4	90.0	90.8	92.4	**96.2**	**97.1**	97.6	90.6	92.4	92.6	92.7
	LMS3	72.0	84.2	86.8	89.2	90.2	94.8	96.5	96.6	91.2	92.3	92.5	92.8
	CASE	72.4	84.2	87.4	89.2	90.3	94.8	96.4	96.9	91.3	92.4	92.7	92.9
	Jev-LDE (ours)	**87.8**	**88.8**	**91.8**	**92.4**	**96.3**	96.1	**97.1**	**97.8**	**91.5**	**92.8**	**93.4**	**93.7**

4.1 Main Results

Jev-LDE improves its Semantic TopK initialization in 46 of 48 classification configurations and achieves the best or tied-best accuracy in 44 (Table 1). Average one-shot accuracy increases from 81.2% to 88.1%, with gains remaining positive on average at larger shot budgets. Improvements are strongest on TREC and under the one-shot constraint, showing that a single membership edit can substantially improve retrieved contexts when the demonstration budget is limited. Beyond this range, one edit improves Semantic TopK in 18 of 24 configurations at 8-shot and 16-shot budgets (Appendix E).

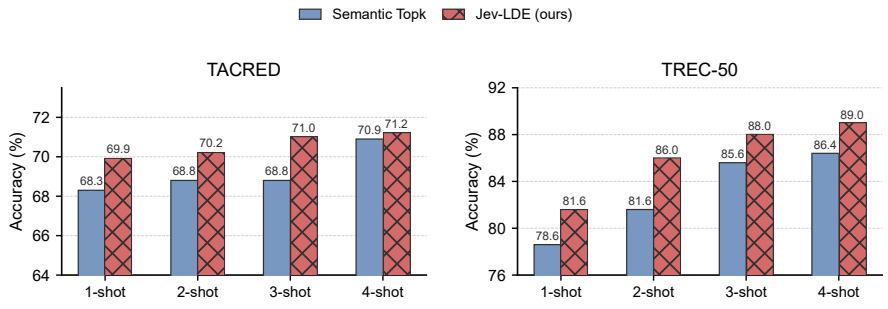


Figure 5: **Evaluation with DeepSeek-V4-Flash.** Accuracy on TACRED and TREC-50 across one to four shots.

The learned policy transfers across target models and benchmarks without additional post-training. Although correctness feedback comes only from Meta-Llama-3-8B-Instruct, Jev-LDE improves re-

trieval in 35 of 36 configurations evaluated with the Qwen targets, 14 of 16 configurations on held-out Banking77, and all displayed settings on held-out SST-2 (Fig. 6, left). Transfer further extends to DeepSeek-V4-Flash on TACRED and TREC-50, with gains at every evaluated shot budget, ranging from 0.3–2.2 and 2.4–4.4 points, respectively (Fig. 5). Together, these results demonstrate that feedback from one frozen target can produce useful edits for new classification settings, including cases where both the benchmark and target model are unseen.

Analysis of Jev-LDE's Transferability. Jev-LDE's demonstration-editing capability may generalize across benchmarks because it judges which examples support a query without solving the task itself. At inference, it selects an action from the query, retrieved demonstrations, and replacement candidates without consulting the target LLM. Multi-benchmark training may encourage reusable judgments of semantic relevance, redundancy, and label distinctions, enabling the same policy to refine contexts for unseen benchmarks. This plausibly explains the observed transfer, although our experiments do not isolate the contributions of these judgments.

Mathematical reasoning shows less consistent benefits. Average accuracy differences on GSM8K remain within one percentage point, while held-out AQuA-RAT accuracy declines by 0.8 and 1.9 points for Qwen2.5-7B-Instruct and Meta-Llama-3-8B-Instruct, respectively (Fig. 6, left; Appendix D). One possible explanation is that scaling compute resources may be more effective than increasing the number of context shots for tasks that require intensive reasoning.

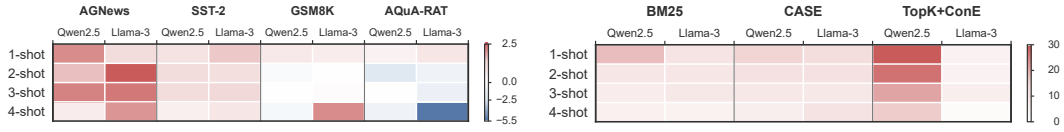


Figure 6: **Additional results and pre-selector analyses.** Left: accuracy differences $\Delta(\mathrm{ACC})$ between Jev-LDE and Semantic TopK on AGNews, SST-2, GSM8K, and AQuA-RAT, Jev-LDE minus Semantic TopK. Right: accuracy differences obtained by applying Jev-LDE to three pre-selectors on TREC.

4.2 Ablation Studies

Pre-selector and Candidate Pool. Jev-LDE improves the performance of initially retrieved set constructed by BM25, CASE, and TopK+ConE in all 24 tested selector–target–shot combinations, using each pre-selector's own initialization and candidate pool (Fig. 6, right). The separate pool-size ablation shows that expanding the available alternatives does not necessarily help: on one-shot TREC, the default $N = 16$ pool achieves 80.6% accuracy, whereas doubling its size reduces accuracy to 79.4% (Fig. 7, right). These results support LDE as a reusable refinement across pre-selectors, with a moderate Replacement neighborhood working best in the pool-size ablation.

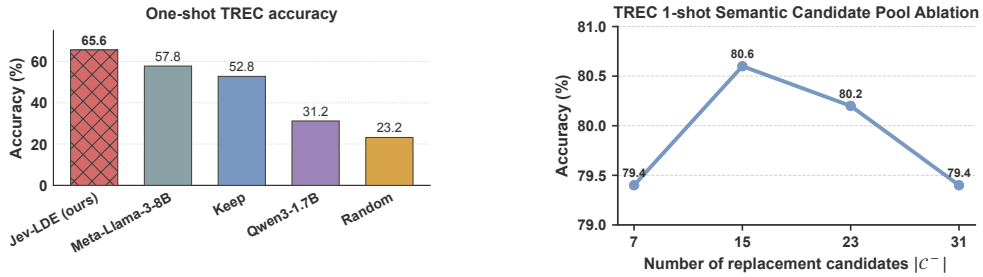


Figure 7: **RLVR and candidate-pool ablations on one-shot TREC.** Left: editing-policy accuracy with Meta-Llama-3-8B-Instruct as the target. Model names denote editors; Random samples a local editing action uniformly. Right: accuracy of Jev-LDE with Qwen2.5-7B-Instruct as a function of the number of replacement candidates.

RLVR's Role in Effective Editing. Fig. 7 compares five editing policies on one-shot TREC under the same evaluation protocols. Jev-LDE reaches 65.6% accuracy, improving over `Keep` by 12.8

points and the directly prompted Meta-Llama-3-8B-Instruct editor by 7.8 points. Random editing and the Qwen3-1.7B initialization achieve 23.2% and 31.2%, respectively, both below `Keep` at 52.8%. At one shot, `Keep` and `Replace` already cover every non-empty singleton choice in the fixed candidate pool (`Delete` reduces to zero-shot at $k = 1$ and is excluded). The improvement therefore demonstrates the value of RL post-training for learning effective action selection within that pool.

Computational Cost. On one-shot TREC with Qwen2.5-7B-Instruct, Jev-LDE raises accuracy from 65.4% to 80.6% with an 11.0% increase in end-to-end evaluation time over Semantic TopK (Fig. 8). Its runtime remains close to LMS3 and CASE and substantially below TopK+ConE. This substantial accuracy gain with modest overhead supports single-edit refinement as an efficient plug to retrieval.

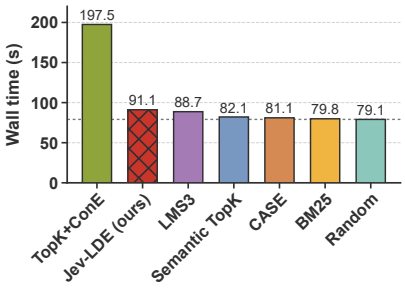


Figure 8: **Computational cost on one-shot TREC.** End-to-end wall time with Qwen2.5-7B-Instruct.

5 Related Work

Retrieval and model-aware utility estimation. Demonstration selection commonly retrieves or ranks examples according to their relevance to the query. Semantic TopK retrieval uses embedding similarity, whereas BM25 relies on lexical matching (Liu et al., 2022; Robertson & Zaragoza, 2009). Demonstration order can also affect predictions even when the selected examples remain unchanged (Lu et al., 2022). Building on retrieval, later methods incorporate model behavior through generator-likelihood supervision, cross-entropy difference, conditional-entropy reranking, model-specific retrieval, and inference stability for mathematical reasoning (Rubin et al., 2022; Iter et al., 2023; Peng et al., 2024; Wang et al., 2024; Liu et al., 2025). These signals make selection more sensitive to the target model, but generally estimate demonstration utility through proxies for the correctness of the complete prompt under the downstream inference protocol.

Target-model feedback and local refinement. Outcome-based methods estimate demonstration utility from downstream predictions. Hashimoto et al. (2024) compare predictions with and without a demonstration to measure incremental utility, while Purohit et al. (2025) explore subsets using bandit rewards. Relevance-Diversity Enhanced Selection learns globally diverse subsets through reinforcement learning (Wang et al., 2025). Difficulty-Stratified Success Prediction trains lightweight judges on offline labels to predict context success (Wang et al., 2026). LDE applies correctness feedback to local post-selection refinement: starting from an existing selector's output, it chooses one `Keep`, `Delete`, or `Replace` action. A frozen target rewards edits according to answer correctness during training; at inference, one edit precedes the final prediction. Compared with global subset search (Purohit et al., 2025; Wang et al., 2025), LDE preserves the retrieved initialization and restricts the decision to local alternatives. It also accommodates multiple successful edits without requiring a unique oracle action. Unlike search-based prompt optimization, which applies local edit operations to instructions and evaluates each result (Prasad et al., 2023; Pryzant et al., 2023), LDE learns a policy that commits to a single edit at inference instead of searching.

6 Conclusion

The purpose of this work is not to pursue the SoTA demonstration selection method; instead, it examines whether a System-1 module can decide how to edit the retrieved demonstration once for more effective ICL. We reformulate this as a local search problem grounded in theoretical analysis and develop the RL post-training pipeline to create Jev-LDE, a lightweight, plug-and-play module that enhances ICL performance with minimal inference overhead. Our classification experiments show that Jev-LDE not only improves ICL performance but also generalizes well to new benchmarks and target models without retraining. These results indicate that learned local editing is a practical method for enhancing few-shot ICL.

AI USE STATEMENT

We used AI tools solely to check for typographical errors and edit inappropriate expressions to improve the manuscript's readability. We did not use AI tools for any other research or writing tasks. We reviewed all AI-assisted edits and take full responsibility for the final content of the paper.

ETHICS STATEMENT

This work evaluates publicly available language benchmarks and pretrained language models and does not involve human participants or private personal data. As an LLM-based editing method, Jev-LDE may inherit biases and errors from its target models and demonstration pools. The method is evaluated for research purposes, and its use in consequential applications would require task-specific validation.

REPRODUCIBILITY STATEMENT

Section 3 specifies the LDE formulation, the action space and post-training procedure of Jev-LDE, and the single-edit inference protocol. Section 4 describes the datasets, baselines, and evaluation protocol. Appendix C provides the editor and downstream prompt templates, Appendix B reports the post-training configuration, and Appendix A documents data processing, decoding settings, and answer normalization. Appendix D provides the complete shot-wise results underlying the additional benchmark and pre-selector panels of Fig. 6, and Appendix E the 8-shot and 16-shot comparison. We will release the complete code of the inference procedure with the post-trained Jev-LDE and all evaluated target LLMs in the final version.

REFERENCES

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In Tsung-Hsien Wen, Asli Celikyilmaz, Zhou Yu, Alexandros Papangelis, Mihail Eric, Anuj Kumar, Iñigo Casanueva, and Rushin Shah (eds.), *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, abs/2501.12948, 2025. URL https://api.semanticscholar.org/CorpusID:284488789.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Bernard N. Grofman, Guillermo Owen, and Scott L. Feld. Thirteen theorems in search of the truth. *Theory and Decision*, 15:261–278, 1983. URL https://api.semanticscholar.org/CorpusID:50576036.

Shivanshu Gupta, Matt Gardner, and Sameer Singh. Coverage-based example selection for in-context learning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 13924–13950, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.930. URL https://aclanthology.org/2023.findings-emnlp.930/.

Kazuma Hashimoto, Karthik Raman, and Michael Bendersky. Take one step at a time to know incremental utility of demonstration: An analysis on reranking for few-shot in-context learning. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3973–3990, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.221. URL https://aclanthology.org/2024.naacl-long.221/.

Dan Iter, Reid Pryzant, Ruochen Xu, Shuohang Wang, Yang Liu, Yichong Xu, and Chenguang Zhu. In-context demonstration selection with cross entropy difference. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 1150–1162, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.81. URL https://aclanthology.org/2023.findings-emnlp.81/.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825/.

Daniel Kahneman. *Thinking, fast and slow*, volume 1. Farrar, straus and Giroux New York, 2011.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pp. 1–7, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1072228.1072378. URL https://doi.org/10.3115/1072228.1072378.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 158–167, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1015. URL https://aclanthology.org/P17-1015/.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In Eneko Agirre, Marianna Apidianaki, and Ivan Vulić (eds.), *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 100–114, Dublin, Ireland and Online, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.deelio-1.10. URL https://aclanthology.org/2022.deelio-1.10/.

Jiayu Liu, Zhenya Huang, Chaokun Wang, Xunpeng Huang, Chengxiang Zhai, and Enhong Chen. What makes in-context learning effective for mathematical reasoning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 38708–38721. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/liu25ac.html.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. URL https://doi.org/10.48550/arXiv.2407.21783.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556/.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 11048–11064, 2022.

Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. Revisiting demonstration selection strategies in in-context learning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9090–9101, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.492. URL https://aclanthology.org/2024.acl-long.492/.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3845–3864, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.277. URL https://aclanthology.org/2023.eacl-main.277/.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.494. URL https://aclanthology.org/2023.emnlp-main.494/.

Kiran Purohit, Venktesh V, Sourangshu Bhattacharya, and Avishek Anand. Sample efficient demonstration selection for in-context learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=cuqvlLBQK6.

Qwen Team. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024. URL https://doi.org/10.48550/arXiv.2412.15115.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, May 2025. URL https://doi.org/10.48550/arXiv.2505.09388.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2655–2671, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.191. URL https://aclanthology.org/2022.naacl-main.191/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. URL https://doi.org/10.48550/arXiv.2402.03300.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/shi23a.html.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170/.

Haochun Wang, Chaofen Yang, Jiatong Liu, Jingbo Wang, Zewen Qiang, Sendong Zhao, Bing Qin, and Ting Liu. Easier to judge than to find: Predicting in-context learning success for demonstration selection. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=iixK49TTop.

Huazheng Wang, Jinming Wu, Haifeng Sun, Zixuan Xia, Daixuan Cheng, Jingyu Wang, Qi Qi, and Jianxin Liao. MDR: Model-specific demonstration retrieval at inference time for in-context learning. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4189–4204, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.235. URL https://aclanthology.org/2024.naacl-long.235/.

Xubin Wang, Jianfei Wu, Yuan Yichen, Deyu Cai, Mingzhe Li, and Weijia Jia. Demonstration selection for in-context learning via reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=sugs65XoGg.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pp. 649–657, Cambridge, MA, USA, 2015. MIT Press.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 35–45, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1004. URL https://aclanthology.org/D17-1004/.

A Dataset Construction and Evaluation Details

We use a unified data and evaluation pipeline to isolate differences among demonstration selectors. All methods use the same evaluation examples, demonstration pools, shot budgets, downstream prompt formats, target-model decoding parameters, and answer-normalization procedures. Candidate scope remains method-specific because global retrieval, semantic reranking, static exemplar selection, and local editing impose different search spaces by design.

Dataset Construction. Each benchmark is converted into a unified ICL representation containing the query, gold label or answer, candidate demonstrations, task metadata, and task type. Demonstrations are drawn exclusively from the corresponding training split, whereas evaluation uses a fixed test split, except for SST-2, whose validation split is used because its public test labels are unavailable. When a training split exceeds the configured demonstration-pool size, examples are sampled deterministically with seed 42. Table 2 summarizes the resulting pools. "Editor Train" indicates whether a benchmark contributes states to Jev-LDE post-training; it does not affect whether its training split can serve as an ICL demonstration pool.

Table 2: **Primary benchmark statistics.** The demonstration pool is drawn from each benchmark's training split. "Editor Train" indicates whether the benchmark contributes Jev-LDE post-training states.

Benchmark	Task	Eval. Split	#Eval.	#Demo Pool	Editor Train
AGNews	Topic classification	Test	1,000	10,000	Yes
TREC	Coarse question classification	Test	500	5,452	Yes
DBPedia	Ontology classification	Test	1,000	10,000	Yes
GSM8K	Mathematical reasoning	Test	1,319	7,473	Yes
SST-2	Sentiment classification	Validation	872	6,921	No
Banking77	Intent classification	Test	3,076	9,993	No
AQuA-RAT	Multiple-choice reasoning	Test	254	10,000	No

Selector-Specific Candidate Scope. All selectors access the same benchmark-specific demonstration pool, but they retain their intended search protocol. Semantic TopK and BM25 retrieve directly from the full pool. TopK+ConE reranks a semantic candidate set, whereas Jev-LDE edits only within a smaller semantic neighborhood. LMS3 and CASE follow their respective global selection protocols rather than being restricted to Jev-LDE's local pool. For CASE, we adapt the original five-cluster, five-shot construction to k clusters for each evaluated shot budget, retaining the paper's settings of $m = 10$, a challenger-set size of 5, 20 validation examples, and $\epsilon = 0.1$. The 20 validation examples used by CASE are drawn from the benchmark-specific demonstration pool and are disjoint from the test split. Table 3 summarizes the candidate construction protocols.

Table 3: **Candidate construction protocols.** "Full" denotes the complete benchmark-specific demonstration pool. For Jev-LDE, the initial k-shot set is retrieved from the full pool via Semantic TopK; the top-16 candidates denote the local neighborhood available to the editor for Replacement candidates.

Method	Search Scope	Working Set	Selection Rule
Random	Global	Full pool	Uniformly sample k demonstrations.
Semantic TopK	Global	Full pool	Rank the full pool by cosine similarity using normalized `BAAI/bge-base-en-v1.5` embeddings, and select the top-k demonstrations.
BM25	Global	Full pool	Select the top-k demonstrations by BM25 score.
TopK+ConE	Two-stage	top-30 candidates	Rerank the top-30 semantic candidates using the conditional-entropycriterion, then choose the top-k demonstrations.
LMS3	Global	Full pool	Combine similarity and stability scores with a 1% similarity filter and no refill.
CASE	Global	Full pool	Construct candidate arms by sampling one demonstration from each of k clusters. Identify the top-m subsets through challenger-arm sampling using validation accuracy as reward, and select the subset with the highest validation accuracy.
Jev-LDE	Local editing	top-16 candidates	Apply one `Keep`, `Delete`, or `Replace` edit to the Semantic TopK initialization.

Prompt and Decoding Protocol. We evaluate Qwen3-8B, Qwen3.5-9B, Qwen2.5-7B-Instruct and Meta-Llama-3-8B-Instruct with $k \in \{1, 2, 3, 4\}$. Downstream generations use vLLM with a maximum model length of 8,192 tokens, temperature 0, and top-$p = 1$. Classification outputs are limited to 8 new tokens, while GSM8K and AQuA-RAT allow up to 1,024 new tokens. The downstream prompts are passed without applying a chat template. The complete target-model prompts are shown in Appendix C. Jev-LDE itself is decoded deterministically with temperature 0 and generates one editor completion, as detailed in Appendix B.

Frontier-Model Evaluation. We additionally evaluate DeepSeek-V4-Flash on TREC-50 and TACRED to assess cross-model transfer in classification settings. The comparison includes Semantic TopK, BM25, and Jev-LDE under shot budgets $k \in \{1, 2, 3, 4\}$. All methods use the same benchmark splits, demonstration pools, target prompts, and API decoding settings within each benchmark.

For TREC-50 (Li & Roth, 2002), we use the fine-grained 50-way question labels. The complete training split of 5,452 examples serves as the demonstration pool, and the complete test split of 500 examples is used for evaluation. Neither split is subsampled.

For TACRED (Zhang et al., 2017), each input contains the sentence together with its designated subject and object entities, and the target output is the corresponding relation label. The demonstration pool contains 10,000 examples sampled without replacement from the training split using label-stratified balanced sampling with seed 42. Evaluation uses 1,000 examples independently sampled from the test split using the same procedure and seed.

DeepSeek-V4-Flash is accessed through the chat-completions API under the model identifier `deepseek-v4-flash`, with thinking disabled. Target decoding is deterministic, using temperature 0, top-$p = 1$, a maximum model length of 4,096 tokens, and at most 32 generated tokens. Although inference uses the chat-completions endpoint, no additional local chat template is applied. Classification prompts follow the template in Appendix C, and the model is instructed to output only the predicted label. Requests are issued with a maximum concurrency of eight and retried up to eight times upon transient API failure.

Answer Normalization and Metrics. For classification, the generated text is mapped to a valid benchmark label before exact-match evaluation. Matching is case-insensitive and ignores articles, punctuation, and repeated whitespace. The evaluator first checks the complete output and its first line for an exact normalized label, then searches for a standalone valid label when additional text is present.

For mathematical reasoning, the evaluator extracts the answer following `####`, the final boxed expression, an explicit "final answer" phrase, or the last numerical expression, in that order. GSM8K answers are compared after removing formatting characters, with numerically equivalent values accepted within an absolute or relative tolerance of 10^{-6}. AQuA-RAT predictions are evaluated by case-insensitive exact match of the extracted option in $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}\}$.

Target-Model Correctness Feedback Ablation. Fig. 7 evaluates five editing policies on one-shot TREC, with Meta-Llama-3-8B-Instruct as the downstream target model. All variants use the same Semantic TopK initialization, top-16 candidate neighborhood, editing state, action space, and downstream evaluation protocol. The prompted editors also share the same editor prompt and output schema. Only the policy used to choose the editing action varies.

Keep evaluates the retrieved demonstration without editing. **Random** uses no decision model and uniformly samples one action from $\mathcal{A}_{\mathrm{LDE}}(\mathcal{Z}_0, \mathcal{C}^-) = \{\texttt{Keep}, \texttt{Delete}(1)\} \cup \{\texttt{Replace}(1, z) : z \in \mathcal{C}^-\}$. With one selected demonstration and 15 Replacement candidates, this space contains 17 actions. The sampled action is applied once before target-model prediction. This differs from the Random selector in Table 3, which samples demonstrations directly from the demonstration pool. **Qwen3-1.7B** uses the editor initialization without correctness-feedback post-training. **Meta-Llama-3-8B-Instruct** is directly prompted to produce the same structured action without editor post-training. **Jev-LDE** uses the Qwen3-1.7B editor after post-training with correctness feedback from the frozen target model. All prompted editors generate one completion and apply at most one edit per query.

Pre-selector Ablation Details. For the pre-selector ablation reported in the right panel of Fig. 6, all experimental settings are identical to those described above, except that the editor is applied to the k-shot set selected by the corresponding method rather than to a Semantic TopK initialization, and the Replacement candidate pool $\mathcal{C}^-$ is the top-16 set produced by that same method. Concretely, for each pre-selector (BM25, CASE, and TopK+ConE) we take its own top-k selection as the initially retrieved set $\mathcal{Z}_0$ and its top-16 candidates as the editing neighborhood, leaving the remaining top-16 candidates as $\mathcal{C}^-$. The editor then applies one `Keep`, `Delete`, or `Replace` action within this method-specific neighborhood. The editor checkpoint, frozen reward target, decoding parameters, and downstream evaluation protocol are unchanged.

Candidate-Pool Size Details. The candidate-pool ablation varies the number of Replacement candidates available to the editor. In this one-shot experiment, the horizontal axis in Fig. 7 reports $|\mathcal{C}^-| \in \{7, 15, 23, 31\}$, corresponding to total semantic candidate-pool sizes $N = |\mathcal{C}| \in \{8, 16, 24, 32\}$ because $|\mathcal{Z}_0| = 1$. In every setting, $\mathcal{C}$ contains the top-N semantic candidates, $\mathcal{Z}_0$ is the Semantic TopK initialization, and $\mathcal{C}^- = \mathcal{C} \setminus \mathcal{Z}_0$. The default is $N = 16$, with 15 Replacement candidates. All other components, including the editor checkpoint, frozen reward target, decoding parameters, and evaluation protocol, are identical to the standard Jev-LDE setting. Results are reported for one-shot TREC with Qwen2.5-7B-Instruct.

Wall-Time Measurement Details. The end-to-end wall-time comparison in the left panel of Fig. 8 uses exactly the inference settings of the one-shot TREC experiment with Qwen2.5-7B-Instruct. Every method is given only the global training pool and the test set; no pre-computed initial demonstration sets or candidate pools are provided. We record the total wall-clock time required to complete the full evaluation, which includes candidate retrieval, demonstration selection or editing, and downstream target-model inference. For Jev-LDE, the measured time therefore covers the Semantic TopK retrieval used to build the initial k-shot set, the construction of the top-16 candidate pool, the editor's single completion, and the downstream target-model evaluation; it does not start from a pre-existing initially retrieved set or candidate pool. Each configuration is run five times, and we report the mean wall time.

Post-Training and Statistical Reporting. Table 1 reports the mean of five end-to-end runs for every method; only Random and CASE sample internally, while the other selectors and Jev-LDE are fixed functions of the shared pool under greedy decoding, so we keep all methods on the same five-run protocol only for uniformity. Every reported number evaluates two frozen artifacts, the post-trained editor and the target models, on fixed splits, so no entry is an average over post-training runs and the comparison never resamples the trainer. Post-training is an artifact-producing step whose tolerance the feedback-complexity bound ties to the amount of correctness feedback rather than to the number of optimization runs, over a deterministic reward, namely binary correctness from a frozen target decoded at temperature 0. Repeating the optimizer would therefore select among policies that this same reward already ranks. The five repeats agree under greedy decoding, so we report no standard deviations and assess significance over evaluation examples instead: each Jev-LDE entry is paired with its own initialization on the same examples, and we read a difference as an effect only when it exceeds the resolution of the split, one example being 0.1 points on the smallest evaluation set, against gaps of 15.2 points on one-shot TREC and 6.9 points on average one-shot accuracy. The same checkpoint improves 46 of 48 configurations, 24 of 24 pre-selector combinations, 14 of 16 on held-out Banking77, and 35 of 36 with the Qwen targets without retraining, which is our check against a favorable draw of one run.

B Jev-LDE Post-Training and Implementation Details

Jev-LDE is initialized from Qwen3-1.7B (Qwen Team, 2025) and post-trained using online correctness feedback from a frozen Meta-Llama-3-8B-Instruct model (Llama Team, 2024). This appendix specifies the construction and filtering of training states, the GRPO configuration, and the resulting training dynamics. Multiple policy rollouts are used only during post-training; evaluation follows the single-sample, single-edit protocol described in Section 3.

Training-State Construction. All post-training queries and candidate demonstrations are drawn exclusively from the benchmark-specific demonstration pools of AGNews, TREC, DBPedia, and

GSM8K, each of which contains only examples from the corresponding training split. No examples or labels from any evaluation split are used for training-state construction, reward-variation filtering, or GRPO post-training. Before retrieval, the training query itself is excluded from its demonstration pool.

Training states are constructed using precomputed demonstration rankings for queries sampled from these pools. For each query, the first k demonstrations form the Semantic TopK initialization, while the remaining demonstrations within the top-16 candidate pool serve as Replacement candidates. We construct states for $k \in \{1, 2, 4, 8, 10\}$, sampling up to $1{,}000$ states for each benchmark–shot-budget pair in each collection round. Each state records the query and gold answer, the selected and candidate demonstration indices, the candidate contents, the shot budget, and benchmark metadata. Demonstrations and queries are each truncated to at most $1{,}000$ characters in the editor input.

State collection is conducted over three rounds. After each round, previously sampled states are excluded using the tuple consisting of the benchmark, split, query identifier, shot budget, and candidate-pool size. The retained states from all rounds are then merged into a single post-training set. No states from the held-out transfer benchmarks are included.

Reward-Variation Filtering. Binary target feedback provides no group-relative signal when all actions available from a state produce the same outcome. We therefore probe each candidate state using a budget of eight valid local actions: `Keep`, up to two distinct `Delete` actions, and up to five distinct `Replace` actions. Every probed edit is applied once and evaluated by the frozen target LLM with temperature 0 and top-$p = 1$. The reward equals 1 when the normalized target prediction matches the gold answer and 0 otherwise.

A state is retained only when its probed actions contain both reward outcomes, equivalently when $\min_a r(a) < \max_a r(a)$. This procedure identifies states for which the local decision is consequential without assigning an optimal action or introducing action labels. The editor must still discover useful actions through policy optimization.

Table 4: **Jev-LDE post-training configuration.** Rollout sampling is used only during post-training.

Component	Setting	Component	Setting
Editor initialization	Qwen3-1.7B	Frozen target LLM	Meta-Llama-3-8B-Instruct
Training benchmarks	AGNews, TREC, DBPedia, GSM8K	Construction shot budgets	$\{1, 2, 4, 8, 10\}$
Candidate-pool size	16	States per benchmark–budget–round	Up to $1{,}000$
Filtering rounds	3	Probe actions	1 `Keep`, up to 2 `Delete`, up to 5 `Replace`
Demo/query truncation	$1{,}000/1{,}000$ characters	Deduplication key	Dataset, split, query, k, pool size
Advantage estimator	Group-relative (`grpo_plus`)	Policy rollouts per state	8
Prompt/generation batch size	$16/16$	PPO mini-batch size	16
Training epochs/updates	$1/752$	Learning rate/weight decay	$10^{-6}/0.1$
Maximum prompt/response length	$8{,}192/1{,}024$ tokens	Gradient clipping	1.0
Actor rollout decoding	Temperature 1.0, top-$p = 1.0$	Reward-target decoding	Temperature 0, top-$p = 1.0$
Clipping margins	0.20 lower, 0.28 upper	Entropy coefficient	0
KL reward/actor loss	Disabled/disabled	Loss aggregation	mean over sequences of per-sequence token means
Number of GPUs	2	Rollout memory utilization	0.35
Checkpoint/evaluation interval	$100/100$ updates	Evaluation checkpoint	Update 752

GRPO Configuration. For every training state, the editor samples eight completions with temperature 1.0 and top-$p = 1.0$. Parsed actions are applied once and scored by the frozen target; invalid outputs receive reward 0. We optimize the editor using the group-relative objective described in Section 3. The target model remains fixed throughout post-training.

Training Dynamics. Fig. 9 reports the two non-redundant optimization trajectories. The logged mean score and mean reward are numerically identical at every update and are therefore represented by a single reward curve. Mean reward increases primarily during the early stage of post-training and remains higher thereafter: its average rises from 0.475 over the first 100 updates to 0.713 over the final 100. Over the same intervals, the actor entropy loss decreases from 0.168 to 0.119. All reported evaluations use the final checkpoint at update 752.

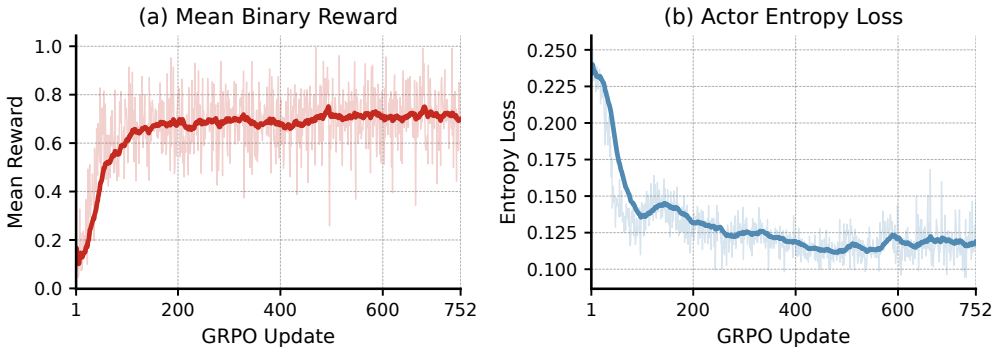


Figure 9: **Jev-LDE post-training dynamics.** Panel (a) reports the mean binary reward returned by the frozen target, and panel (b) reports the actor entropy loss. Light curves show the raw per-update values; solid curves apply the debiased exponential moving-average smoother used by the reference plotting protocol, with smoothing parameter 0.5.

C Prompt Templates and Examples

Fig. 10–13 present the prompt templates used by the downstream target models and Jev-LDE. Braced fields denote task-dependent content inserted at runtime, and ellipses denote repeated demonstrations or labels. The editor uses the same prompt structure during post-training and evaluation, while all demonstration-selection methods share the same downstream prompts. The mathematical-reasoning formats follow the evaluation protocols of LMS3 and CASE (Liu et al., 2025; Purohit et al., 2025).

Classification target prompt

Prompt Template:
You are given examples of a classification task.
{task instruction, if available}
Output only the predicted label.

Input: {demonstration input 1}
Label: {demonstration label 1}

...

Input: {query}
Label:

Figure 10: **Classification target-model prompt template.**

GSM8K target prompt

Prompt Template:
Here are some examples you can refer to.

Q: {demonstration question 1}
A: {demonstration solution 1}

...

---Now it's your turn!
- Take a deep breath
- Think step by step
- I will tip $200

Q: {query}
A:

Figure 11: **GSM8K target-model prompt template.**

AQuA-RAT target prompt

Prompt Template:
Instruction: You are a helpful, respectful and honest assistant helping solve math word tasks requiring reasoning. Follow given examples and solve the tasks in step by step manner.

Exemplars:
[Question]: {demonstration question 1}
[Options]: {demonstration options 1}
[Explanation]: {demonstration rationale 1}
[Answer]: The option is {demonstration answer 1}

...

Test Input:
[Question]: {query}
[Options]: {query options}
[Explanation]:
[Answer]:

Figure 12: **AQuA-RAT target-model prompt template.**

Jev-LDE editor prompt

Prompt Template:
You are a demonstration selection policy for in-context learning.

Goal: choose one local edit to the current demonstrations so that a target language model is more likely to answer the query correctly.
Do not answer the query yourself. Judge which demonstrations will help the target model.

Target task:
{task instruction}

Selection criteria:
- Prefer demonstrations that match the query's task type, reasoning pattern, label boundary, and answer format.
- Avoid demonstrations that are only superficially similar but use a misleading label, solution pattern, or answer style.
- Avoid redundant selected demonstrations when a candidate adds clearer coverage.
{task-specific selection rules}

Available actions:
- Keep: use when the selected demonstrations are already strong and no candidate is clearly better.
- Delete: remove one selected demonstration if it is irrelevant, redundant, misleading, or too long.
- Replace: replace the weakest selected demonstration with a clearly more helpful candidate.

Output rules:
- You may reason inside <think> </think> tags.
- Put the final action JSON inside <answer> </answer> tags.
- The reward parser uses only the final <answer> content; do not put the final action only in <think>.
- Do not include a reason field in the final JSON.

Query to support:
{query}

Current selected demonstrations:
[S1]
{selected demonstration 1}
...
[Sk]
{selected demonstration k}

Candidate demonstrations:
[C1]
{candidate demonstration 1}
...
[Cm]
{candidate demonstration m}

Allowed JSON formats:
<answer>{"action":"Keep"}</answer>
<answer>{"action":"Delete","selected_id":"S1"}</answer>
<answer>{"action":"Replace","selected_id":"S1","candidate_id":"C1"}</answer>

Figure 13: **Jev-LDE editor prompt template.**

D Completed Results for Heatmap

Table 5: **Accuracy values underlying the generalization heatmap (Fig. 6, left).** Semantic TopK and Jev-LDE accuracy (%) on AGNews, SST-2, GSM8K, and AQuA-RAT for Qwen2.5-7B-Instruct and Meta-Llama-3-8B-Instruct. "Mean Δ" is the four-shot average difference of Jev-LDE relative to Semantic TopK.

Dataset	Target	Method	1-shot	2-shot	3-shot	4-shot	Mean Δ
AGNews	Qwen2.5-7B	Semantic TopK	85.9	87.9	88.7	88.9	
		Jev-LDE	87.9	89.4	90.8	89.5	+1.6
	Llama-3-8B	Semantic TopK	83.8	86.5	87.2	87.9	
		Jev-LDE	84.9	89.0	89.4	89.8	+1.9
SST-2	Qwen2.5-7B	Semantic TopK	91.4	92.3	91.9	93.4	
		Jev-LDE	92.3	93.4	93.0	93.9	+0.9
	Llama-3-8B	Semantic TopK	90.6	91.4	91.9	91.9	
		Jev-LDE	92.0	92.4	93.1	92.7	+1.1
GSM8K	Qwen2.5-7B	Semantic TopK	86.6	88.9	88.1	89.0	
		Jev-LDE	87.3	88.4	88.1	88.3	−0.1
	Llama-3-8B	Semantic TopK	72.4	74.2	73.4	71.7	
		Jev-LDE	73.0	74.2	73.5	73.7	+0.7
AQuA-RAT	Qwen2.5-7B	Semantic TopK	62.6	75.6	74.8	76.8	
		Jev-LDE	63.0	73.2	74.8	75.6	−0.8
	Llama-3-8B	Semantic TopK	31.1	33.1	35.0	36.2	
		Jev-LDE	31.9	31.9	33.5	30.7	−1.9

Table 6: **TREC pre-selector ablation (%).** The editor is applied to the top-16 pool of each pre-selector on TREC. "Unedited" is the selector's own result; "+Jev-LDE" applies one local edit. Δ is the difference relative to "Unedited".

Target	Variant	1-shot	2-shot	3-shot	4-shot	Mean Δ
Qwen2.5-7B	BM25	64.6	77.6	81.6	84.4	
	BM25 +Jev-LDE	82.2	86.8	88.6	89.8	+9.8
	CASE	65.6	75.4	79.2	81.8	
	CASE +Jev-LDE	80.4	86.6	87.6	88.0	+10.2
	TopK+ConE	46.6	57.2	60.6	65.6	
	TopK+ConE +Jev-LDE	75.6	83.4	81.0	81.4	+22.9
Llama-3-8B	BM25	45.6	68.4	69.0	70.6	
	BM25 +Jev-LDE	55.4	77.6	77.4	75.6	+8.1
	CASE	52.6	72.0	72.6	71.2	
	CASE +Jev-LDE	65.2	83.0	80.8	81.8	+10.6
	TopK+ConE	55.2	72.2	72.0	74.0	
	TopK+ConE +Jev-LDE	60.0	77.0	78.2	75.8	+4.4

E Classification Results at Larger Shot Budgets

We follow the experimental settings of the main classification experiments, except that we increase the initial shot budget to $k \in \{8, 16\}$ and use a semantic top-30 candidate pool for Jev-LDE. Table 7 compares Semantic TopK with its locally edited version on TREC, Banking77, and DBPedia.

Table 7: **8-shot and 16-shot classification accuracy (%).** TopK denotes Semantic TopK, and +LDE applies one local edit using Jev-LDE. Bold indicates the better result within each pair, including ties. Δ denotes the accuracy difference in percentage points, computed from the displayed values.

Target	Dataset	8-shot			16-shot		
		TopK	+LDE	Δ	TopK	+LDE	Δ
Qwen2.5-7B	TREC	83.00	**84.00**	+1.00	84.60	**85.20**	+0.60
	Banking77	91.84	**92.04**	+0.20	**92.17**	91.97	−0.20
	DBPedia	**98.00**	97.70	−0.30	98.10	**98.40**	+0.30
Llama-3-8B	TREC	68.80	**74.80**	+6.00	52.60	**54.80**	+2.20
	Banking77	**90.90**	90.51	−0.39	62.00	**62.68**	+0.68
	DBPedia	72.50	**76.60**	+4.10	61.70	**66.20**	+4.50
Qwen3-8B	TREC	88.20	**89.20**	+1.00	89.80	**91.60**	+1.80
	Banking77	92.39	**92.78**	+0.39	**92.65**	92.56	−0.09
	DBPedia	97.90	**98.00**	+0.10	98.40	**98.60**	+0.20
Qwen3.5-9B	TREC	**93.00**	**93.00**	0.00	95.00	**95.40**	+0.40
	Banking77	92.72	**93.01**	+0.29	**93.27**	93.24	−0.03
	DBPedia	98.10	**98.20**	+0.10	98.10	**98.40**	+0.30

Jev-LDE improves accuracy in 18 of 24 configurations and matches Semantic TopK in one. Gains are largest on Llama-3-8B, reaching 6.00 percentage points on 8-shot TREC and 4.50 points on 16-shot DBPedia. The five decreases are small, ranging from 0.03 to 0.39 points. These results show that a single local edit remains effective at larger shot budgets across the evaluated targets and benchmarks.

F Proofs of the Theoretical Results

F.1 Proof of Proposition 1

Conditional reliability. Index candidates by retrieval rank, with the first k in $\mathcal{Z}_0$. Let $v \in \{0,1\}^N$ indicate whether each candidate's query-label judgment is correct. Probabilities refer to this ranked vector over random evaluation states, without conditioning on the full realized query and demonstration contents. For a retrieved position with $v_i = 0$, let v^{+i} change only that position to one. The assumption implies $\Pr(v) \le [\frac{1-\alpha}{\alpha}] \Pr(v^{+i})$. This also holds for zero-mass configurations and requires no independence or full-support assumption.

Minimum repair distance. Write $k = 2m + 1$. Let B count incorrect retrieved judgments and H count correct judgments in $\mathcal{C}^-$. Retrieval fails when $B \ge m + 1$. Each Replacement adds at most one correct judgment, so repair requires at least $B - m$ Replacements. This is achievable exactly when $H \ge B - m$, by replacing incorrect judgments with correct ones. The same condition characterizes global repairability: the pool must contain at least $m + 1$ correct judgments. Thus $D = B - m$ on $0 < D < \infty$.

Counting feasible configurations. For $b = m + 1, \dots, k$, define $w_b = \Pr(B = b, H \ge b - m)$. Each configuration in w_{b+1} has $b + 1$ incorrect retrieved positions. Flipping any one to correct yields a configuration in w_b, preserving global feasibility. A configuration in w_b has at most $k - b$ such predecessors. Summing the conditional reliability inequality over these edges gives, for $b = m + 1, \dots, k - 1$,

$$(b + 1)w_{b+1} \le (k - b)\frac{1 - \alpha}{\alpha} w_b.$$

The reverse count is an upper bound because reversing a flip may destroy feasibility. No reliability assumption is imposed after conditioning on feasibility.

Since $\frac{k-b}{b+1} \leq 1$ in this range, iteration gives $w_{b+h} \leq [\frac{1-\alpha}{\alpha}]^h w_b$. Set $w_b = 0$ for $b > k$. Therefore,

$$\Pr(D > h | 0 < D < \infty) = \frac{\sum_{b=m+1}^{k} w_{b+h}}{\sum_{b=m+1}^{k} w_b} \leq \left(\frac{1-\alpha}{\alpha}\right)^h.$$

The denominator is positive by assumption. For $h \geq m+1$ the numerator is zero, including every $h \geq 1$ when $k = 1$.

Refinement and scope. The sharper ratio $\frac{k-b}{b+1} \leq \frac{k-1}{k+3}$ gives the finite-shot tail bound $[\frac{(k-1)(1-\alpha)}{(k+3)\alpha}]^h$. Setting $h = 1$ gives coverage at least $1 - \frac{(k-1)(1-\alpha)}{(k+3)\alpha}$. Adding `Delete` cannot reduce coverage of the fixed-shot comparator. Conditional reliability permits dependence: judgments drawn independently given a shared reliability in $[\alpha, 1]$ remain conditionally reliable after that latent variable is marginalized. Marginal accuracy alone is insufficient: five retrieved judgments that are jointly correct with probability 0.99 and jointly wrong otherwise require three Replacements whenever retrieval fails, even with abundant correct candidates. The result is specific to fixed binary judgments and majority aggregation, rather than arbitrary multiclass or context-dependent predictions.

F.2 Proof of Proposition 2

Reward estimation. For fixed N, k, the action count $|\mathcal{A}_{\mathrm{LDE}}| = 1 + k + k(N-k)$ is constant across states. For any fixed $\boldsymbol{\phi} \in \mathcal{P}$, the independent variables $|\mathcal{A}_{\mathrm{LDE}}| r_i \pi_{\boldsymbol{\phi}}(a_i|s_i)$ lie between zero and $|\mathcal{A}_{\mathrm{LDE}}|$, and their mean equals the policy's population reward under uniform exploration. Their conditional second moment is at most $|\mathcal{A}_{\mathrm{LDE}}| \sum_a \pi_{\boldsymbol{\phi}}(a|s_i)^2 \leq |\mathcal{A}_{\mathrm{LDE}}|$, since $r_i^2 \leq 1$ and action probabilities sum to one. The unconditional variance is therefore also at most $|\mathcal{A}_{\mathrm{LDE}}|$.

Uniform accuracy and policy selection. Bernstein's inequality bounds the probability that the sample average differs from its population mean by more than $\frac{\varepsilon}{2}$ by

$$2\exp\left(-\frac{|\mathcal{D}|\varepsilon^2}{16|\mathcal{A}_{\mathrm{LDE}}|}\right) \leq \frac{\delta}{|\mathcal{P}|}.$$

Here Bernstein's denominator is bounded using $8 + \frac{4\varepsilon}{3} \leq 16$ for $0 < \varepsilon < 1$, and the last inequality uses the stated sample-size condition. A union bound thus gives estimation error at most $\frac{\varepsilon}{2}$ for every policy simultaneously, with probability at least $1 - \delta$. Multiplying the empirical objective in Proposition 2 by $\frac{|\mathcal{A}_{\mathrm{LDE}}|}{|\mathcal{D}|}$ gives exactly this sample average without changing its maximizer. Comparing the empirical maximizer with the population maximizer incurs at most two estimation errors, hence an expected reward gap at most ε. No assumption that the policy class contains an oracle is needed.

Action-space comparison. The proof depends on action count, so it also applies to other action families. For fixed-shot sets, `Keep`/`Replace` has $1 + k(N-k)$ actions, compared with $\binom{N}{k}$ globally. Including `Delete` gives $1 + k + k(N-k)$ local actions versus $\binom{N}{k} + \binom{N}{k-1}$ global sets of size k or $k-1$: 53 versus 2,380 when $N = 16, k = 4$. For order-sensitive contexts, the global counts become $\frac{N!}{(N-k)!}$ and $\frac{N!}{(N-k)!} + \frac{N!}{(N-k+1)!}$, respectively; these families contain the corresponding local prompts. Sorting global subsets canonically need not preserve that inclusion. With policy complexity and exploration coverage held fixed, the reduction in action count translates into a smaller sufficient feedback bound. This provides a learning-theoretic rationale for local editing relative to global subset selection. For fixed k, the global set count is polynomial in N, and the local and global families coincide at $k = 1$.

Learning assumptions. The policy sees s_i, not the gold label. Its action probabilities aggregate all completions producing each valid action. If invalid outputs are mapped to a fallback, both their probabilities and rewards must be handled consistently. Multiple rollouts on one query are not additional independent queries. The data-independent finite class and evaluation-distribution sampling are assumptions of the benchmark, not properties established for reward-filtered GRPO training. With exploration probability at least $\frac{\gamma}{|\mathcal{A}_{\mathrm{LDE}}|}$, where $0 < \gamma \leq 1$, importance weighting gives the same bound with action factor $\frac{|\mathcal{A}_{\mathrm{LDE}}|}{\gamma}$. The conclusion concerns sampled-action expected reward

on the same distribution. A finite class of greedy decoders can be analyzed instead using action indicators, but the sampled-policy guarantee alone does not imply greedy performance or transfer across distributions.

G LIMITATIONS

Task coverage and transfer. Our strongest results concern classification tasks. Improvements on GSM8K are small and inconsistent, while average accuracy decreases on held-out AQuA-RAT. The observed transfer across classification benchmarks and target models therefore does not establish comparable benefits for mathematical reasoning or broader generation tasks.

Scope of local refinement. LDE can retain the initial demonstration set, delete one example, or replace one example within the available candidate pool. It cannot recover useful demonstrations outside that pool or address cases requiring multiple coordinated changes. Its effectiveness therefore depends on the initial retrieval and the available replacement candidates.